Line Quadtrees: A Hierarchical Data Structure for Encoding Boundaries

Hanan Samet
Robert E. Webber

Computer Science Department
and
Computer Vision Laboratory
University of Maryland
College Park, MD 20742

ABSTRACT

A new data structure termed a line quadtree is presented that encodes both a region and its boundary in a hierarchical manner. It is similar to the quadtree with the modification that with each node information is stored as to which of its sides are adjacent to a boundary. The information is available for both terminal and non-terminal nodes. Use of such a data structure is claimed to facilitate a number of basic region processing algorithms including boundary following and map superposition.

1. Introduction

Region representation plays an important role in image processing and geographic information systems. Data representations in use include point by point representations such as binary arrays and run length coding, vector representations such as chain codes [4], and hierarchical representations such as quadtrees [7] and strip trees [1].

In this paper, we address the issue of hierarchically representing images which are segmented into a number of different regions rather than two (foreground and background) as is done in work relating to quadtrees. We introduce a new data structure termed a "line quadtree" that encodes both the region and its border in a hierarchical manner. This is in contrast to the conventional quadtree which only encodes areas in a hierarchical way and to the strip tree [1] which only encodes curves in a hierarchical manner. Our presentation is an iterative one that demonstrates the various design decisions which led to our adoption of a particular definition for the data structure.

2. Line Quadtrees

Geometry populates the plane with three distinct entities - the point, the line, and the region. Finkel and Bentley [3] apply the quadtree data structure to the problem of storing a set of points. Klinger [7] (also [8,9]) used quadtrees to store regions (henceforth called region quadtrees). Subsequent work included applications to graphics [5,6] as well as interconversions with other representations such as chain codes [2,11], rasters [12,13] and binary arrays [14]. Herein, we apply the quadtree data structure to the problem of storing line segments that partition a plane.

We represent the partition of a plane as a two color quadtree where the boundary lines become narrow BLACK regions and the remainder of the map is colored WHITE. This is awkward for two reasons: (1) narrow regions are costly in terms of the number of nodes in the quadtree, and (2) it commits the user to a specific thickness of the boundary line which may be unfortunate when outputting the map.

The bottom-up definition of a region quadtree starts with planar fragments of uniform color, called pixels. The bottom-up definition of a line quadtree, the data structure proposed herein, starts with pixels that record the presence or absence of an edge on each of the four sides of a pixel. The line quadtree is subsequently built by merging the nodes of a complete 4-ary tree where each leaf node corresponds to a pixel.

In order to complete the bottom-up definition of a line quadtree, we must give a compatibility criteria for merging. The scheme that we propose is called the propagated criteria of weak-formedness. This criteria and its immplementation are described as the end result in a successive pattern of refinement from the criteria of identicality (algorithm 1) to the criteria of strong-formedness (algorithm 2) to the criteria of weak-formedness (algorithm 3) to the propagated criteria of weak-formedness (algorithm 4). The implementations can be found in [16].

The intuitions involved in line quadtrees are best motivated graphically. Figure 1 shows the standard arrangement of boundaries and quadrants. The four boundaries/ sides/ directions/ edges are labeled N, E, S, W. A predicate EDGE(node, side) is true (hence marked SET as opposed to CLEAR) iff the specified side of the node is adjacent to the boundary of the region of which the node is a member. In Figure 2, the bold lines indicate the region's boundaries and the light lines indicate the digitization. All maps are surrounded by a border to maintain the separation of regions when the maps undergo linear transformations. In Figure 3, the complete 4-ary tree that encodes the map M is shown. Each square represents a node in the tree. The four largest squares inside a given square represent the four sons of that node. An empty square represents a leaf. The outermost square represents the tree's root. The EDGE information for a node is stored in the picture by using a bold line on any side whose EDGE value is marked SET and otherwise using a light line. Note that algorithms 1 through 3, below, do not use the EDGE information of interior nodes and thus the interior nodes are drawn under the assumption that all EDGE information is marked CLEAR.

Algorithm 1, embodies the criteria of identicality, i.e., four brothers are merged (and their father is replaced by one of his sons) iff they contain identical information in their corresponding EDGE fields. Application of algorithm 1 to the 4-ary tree in Figure 3 produces the quadtree of Figure 4. Note that only two 4-tuples of brothers were merged.

Although the criteria of identicality is adequate for region quadtrees, it is inadequate for line quadtrees. Nodes that border edges tend to have as brothers nodes that do not border edges on the same corresponding side. This can be seen by observing the example map in Figure 3.

Ideally, the merging criteria should maximize the probability that it will be met by many nodes of the tree representation of a "typical" image. On the other hand, the allowable merging criteria must permit the reconstruction of the original digitized image from the resulting tree.

We will not prove the optimality of our criteria with respect to the above restrictions. Instead, we will compare our criteria to a region quadtree where each region is given a separate color and borders are represented implicitly by the two squares of different color that the border separates. For Figure 2, such a region quadtree would use six colors as is shown in Figure 8.

Next we consider the criteria of strong-formedness (implemented by algorithm 2) stated as follows. If the submap represented by a given subtree corresponds to a square with zero or more entire sides missing, then that subtree can be replaced by a single leaf. Since there are no lines crossing the interior of a drawn square, it follows that a single leaf can never represent a submap that contains more than one region. Figure 5 presents the quadtree resulting from applying this merging condition to the 4-ary tree in Figure 3. Node alpha in Figures 3 and 5 shows the result of the patterns of four brothers being merged to form one pattern. In the NW son of the root of Figure

82CH1761-6/82/0000/0090$00.75 © 1982 IEEE

3, we see the result of this merging process having been performed on two separate levels. Node beta in Figures 3, 4, and 5 illustrates four sons that could not be merged because they had edges interior to their non-square pattern (although they were merged in algorithm 1). Node gamma of Figure 5 shows four sons that could not be merged because the S border of gamma's SW son is marked SET but that of gamma's SE son is marked CLEAR; so that together, they form only a partial border.

Although the number of nodes in Figure 5 is considerably smaller than in Figure 4, we still have not merged all nodes that would be merged by the six color region quadtrees of Figure 2. In particular, note that the SE son of the root in Figure 5 resides entirely inside region M6 of Figure 2 (and hence represented by one node of color M6), whereas in Figure 5 this node has four offspring.

In order to be able to merge the above mentioned regions, we weaken the criteria of strong-formedness to allow two different edge values to be combined but still to provide for reconstruction of the original map. This criteria of weak-formedness (implemented by algorithm 3) states that four brothers are merged if there are no "inner" edges in the resulting submap and the resulting node has the "logical and" of the EDGE information.

Figure 6 shows the result of applying algorithm 3 to the complete 4-ary tree of Figure 3. Note that the SE son of the root is now a leaf and also that merging occurred in both the NW son and the SE son of the SW son of the root. Indeed all the hoped-for mergers occurred.

Two items are worthy of further note. First, that if four nodes fail to meet the criteria of weak-formedness, then they must encode at least two different regions. In such a case, the region quadtree would not have merged these brothers either. Therefore, the number of nodes in the line quadtree is bounded from above by the number of nodes in the corresponding region quadtree. Analogous reasoning about the significance of the absence of inner edges leads to the conclusion that the number of nodes in the line quadtree is equal to the number of nodes in the corresponding region quadtree. Second, it can be shown [16] that it is fairly simple to reconstruct the digitized image from its line quadtree using the weak-formedness criteria.

Until now, we have been silent as to what information is stored in internal (i.e., GRAY) nodes. By propagating edge information upward in the tree we can speed up edge-following algorithms (i.e., the quadtree to boundary code transformation). In particular, an edge is propagated upward as marked SET iff at the higher level, it could be followed without reference to the lower nodes that form it, e.g., the south side of M6 in Figure 2. Hence we propagate the "logical and" of the subtended edges of the sons to be the value of the interior node's edges. Figure 7 illustrates the result of doing this to the quadtree of Figure 6. This is termed the "propagated criteria of weak-formedness" (encoded by algorithm 4).

The edge-following algorithms are straight forward extensions of the quadtree-to-boundary-code algorithm. They run slightly faster because they can sometimes move onward on the basis of information stored at an interior node and do not have to examine all the terminal nodes along the path. For example, an edge following algorithm processing the bottom edge of the map of Figure 2, with the aid of the quadtree of Figure 7, would be able to use the fact that the S side of the root is solid and thereby never have to descend to the SE son of the SW son of the SW son of the root to verify the presence of that segment of the edge.

3. Concluding Remarks

We have presented a data structure for storing maps and their boundaries in a hierarchical manner without excessive waste of storage or having to solve the messy problem of graph coloring. Such coloring would be necessary in order to use region quadtree algorithms of comparable storage frugality. The coloring could be achieved by use of connected component labeling algorithms [15] followed by the linear-time five coloring algorithm of [10] (hence minimizing the number of bits needed for colors). However, this entails a substantial loss of information stored in the internal nodes that can be used by line-following algorithms.

Our approach is predicated on the desire to be able to determine and represent in a hierarchical manner both the areas and the borders of the regions comprising the maps. Thus our techniques are more applicable to a decomposition of a map into counties, states, etc. rather than contour lines, point data such as cities, or roads and rivers, which could be better represented by use of other data structures such as point space quadtrees [3] for cities and strip trees [1] for roads and rivers.

We have seen how the line quadtree is used to facilitate border following algorithms. In [16] it is shown that the line quadtree can also be used to speed up the more traditional postorder tree traversal algorithms such as superposition of one map on another.

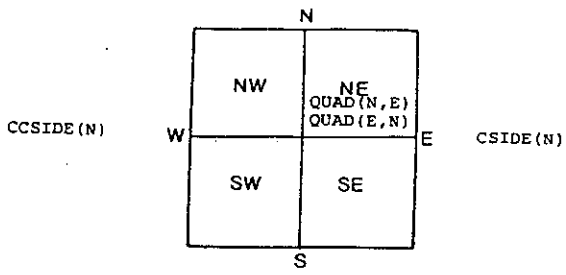

Figure 1. Quadrants and Boundaries Labeled

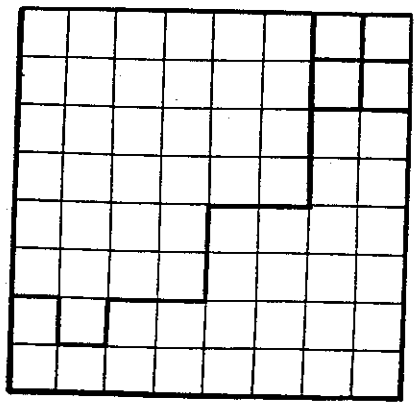

Figure 2. Picture (Map) M on 8 x 8 array of pixels

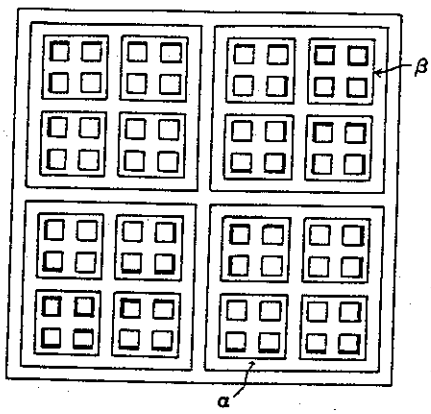

Figure 3. Complete 4-ary tree for Map M

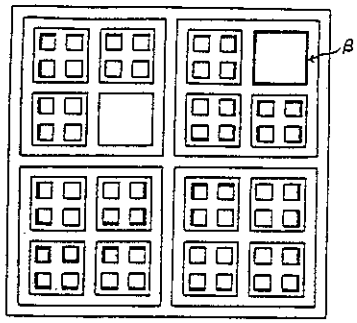

Figure 4. The quadtree produced by Algorithm 1

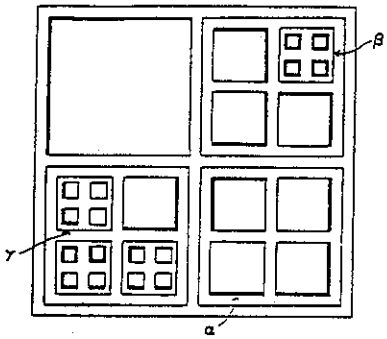

Figure 5. The quadtree produced by Algorithm 2

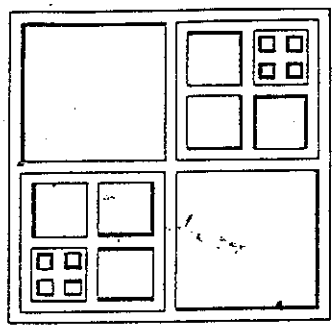

Figure 6. The quadtree produced by Algorithm 3

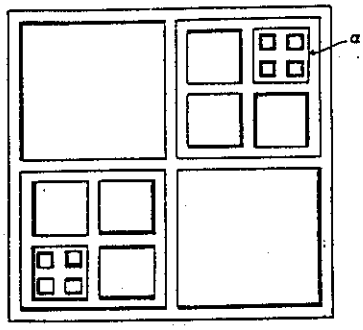

Figure 7. The quadtree produced by Algorithm 4

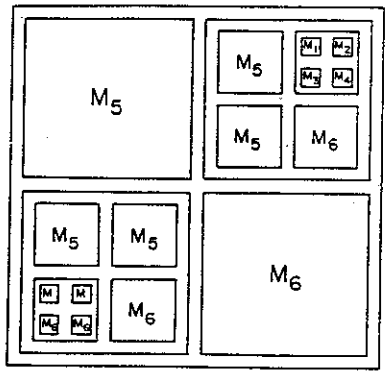

Figure 8. The 6 color quadtree for M

4. References

1. D. H. Ballard, Strip trees: a hierarchical representation for curves, *Communications of the ACM*, May 1981, 310 - 321.
2. C. R. Dyer, A. Rosenfeld, and H. Samet, Region representation: boundary codes from quadtrees, *Communications of the ACM*, March 1980, 171 - 178.
3. R. A. Finkel and J. L. Bentley, Quadtrees: a data structure for retrieval on composite keys, *Acta Informatica 4*, 1974, 1 - 9.
4. H. Freeman, Computer processing of line-drawing images, *ACM Computing Surveys 6*, 1974, 57 - 97.
5. G. M. Hunter and K. Steiglitz, Operations on images using quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 1979, 145 - 153.
6. G. M. Hunter and K. Steiglitz, Linear transformations of pictures represented by quadtrees, *Computer Graphics and Image Processing 10*, 1979, 289 - 296.
7. A. Klinger, Patterns and search statistics, in *Optimizing Methods in Statistics*, J. S. Rustagi (Ed.), Academic Press, New York, 1971.
8. A. Klinger and C. R. Dyer, Experiments in picture representation using regular decomposition, *Computer Graphics and Image Processing 5*, 1976, 68 - 105.
9. A. Klinger and M. L. Rhodes, Organization and access of image data by areas, *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*, 1979, 50 - 60.
10. D. Matula, Y. Shiloach, and R. Tarjan, Two linear-time algorithms for five coloring a planar graph, Stanford Technical Report STAN-CS-80-830, November 1980.
11. H. Samet, Region representation: quadtrees from boundary codes, *Communications of the ACM*, March 1980, 163 - 170.
12. Samet, An algorithm for converting rasters to quadtrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 1981, 93 - 95.
13. H. Samet, Algorithms for the conversion of quadtrees to rasters, Computer Science TR-979, University of Maryland, College Park, November 1980.
14. H. Samet, Region representation: quadtrees from binary arrays, *Computer Graphics and Image Processing 13*, 1980, 88 - 93.
15. H. Samet, Connected component labeling using quadtrees, *Journal of the ACM*, July 1981, 487 - 501.
16. H. Samet and R. E. Webber, On encoding boundaries with quadtrees, Computer Science TR-1128, University of Maryland, College Park, MD, December 1981.